Учреждение образования

“Белорусский государственный университет информатики и радиоэлектроники”

Кафедра электронных вычислительных машин

Лабораторная работа №2

ПАКЕТНАЯ ПЕРЕДАЧА ДАННЫХ

Проверил:

Глоба А.А.

Выполнил: ст. гр. 850503

Осетник Д.А.

Минск, 2020

Для именования порции информации, передаваемой по каналам компьютерных (и не только компьютерных) сетей, используется обобщенный термин пакет (packet). Пакет содержит последовательно сформированные станцией-передатчиком поля (fields), предназначенные для их интерпретации в станции-приемнике. В общем случае, пакеты могут быть самыми разнообразными (как по структуре, так и по длине).

Типовая структура пакета включает в себя:

1. Флаг начала пакета – позволяет определить начало пакета.
2. Адрес назначения – позволяет указать станцию, для которой предназначен пакет.
3. Адрес источника – позволяет указать станцию, из которой был отправлен пакет.
4. Прочие поля – специфические поля определенной реализации.
5. Данные – полезная нагрузка пакета.
6. Контрольная сумма – позволяет проверить был ли искажен пакет при передаче (контрольная сумма не всегда может определить искажение пакета, например бит четности ошибочно примет двойную ошибку за корректные данные).

Часть пакета, которая включает поля, расположенные до начала данных, называют заголовком, а поля, расположенные после данных хвостовиком.

При продвижении информации между уровнями возникает необходимость в преобразованиях структур данных. Преобразования выражаются в инкапсуляции и декапсуляции. Под инкапсуляцией понимается вкладывание пакета определенного вышестоящего уровня в поле данных пакета смежного нижестоящего уровня в процессе подготовки к передаче, то есть при продвижении сверху вниз. Под декапсуляцией понимается обратное действие после приема, то есть при продвижении снизу вверх.

Для правильно интерпретации пакета его необходимо полностью считать из канала, с соблюдением порядка чтения пакета. Так как станция-приемник может подключиться к каналу в произвольный момент времени, возникает проблема с определение флага начала пакета. Так как флаг начала пакета является зарезервированной последовательностью, такая же последовательность может появиться в пакете в качестве информации, передаваемой в полезной нагрузке пакета.

Для решения данной проблемы используются действия, изменяющие следующую за флагом цифровую последовательность. Они называются битстаффинг и байтстаффинг в бит-ориентированных и байт-ориентированных системах соответственно.

При битстаффинге совпадающая с начальным флагом последовательность разбивается при помощи вставки бита с соответствующим значением. Классическим флагом начала пакета является флаг со значение 01111110. Оптимальная реализация битстаффинга заключается в вставке после нуля и шести единиц седьмой единицы на передатчике, а на приемнике ее удаление. Обычно реализуется аппаратно.

При байтстаффинге происходит манипуляция байтами, а не битами. Это облечает работу при программировании (позволяет избегать битовых операций), но является более затратным. Решением проблемы при байт-стаффинге является использование ESC-символов, которые сообщают о факте замены.

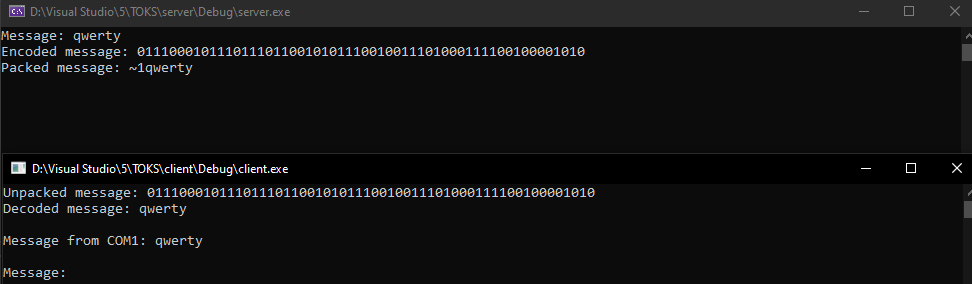
Алгоритм работы программы при отправке сообщения:

1. Сообщение переводится в двоичное представление
2. При нахождении последовательности “111111” добавляется “1”, поиск последовательности продолжается с места вставки “1”. Таким образом в последовательность, например из 7 единиц, добавится только одна единица.
3. Формируется пакет с 8 байтами полезной нагрузки.
4. Сообщение отправляется.

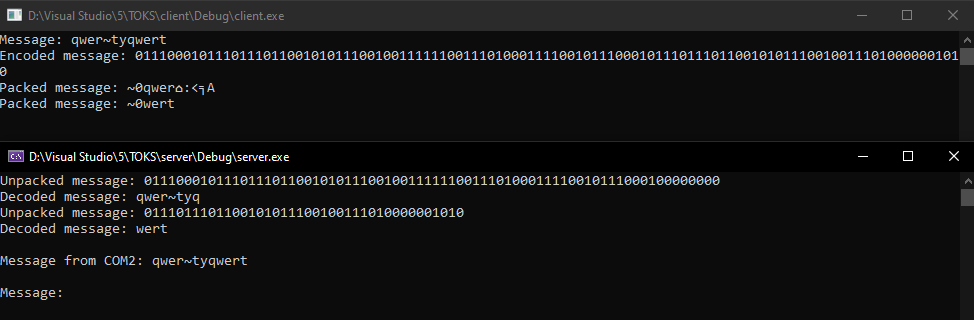
Алгоритм работы при получении сообщения:

1. Сообщение переводится в двоичное представление.
2. При нахождении последовательности “1111111” удаляется “1”, поиск продолжается с места удаления “1”.
3. Пакеты распаковываются и формируется сообщение.

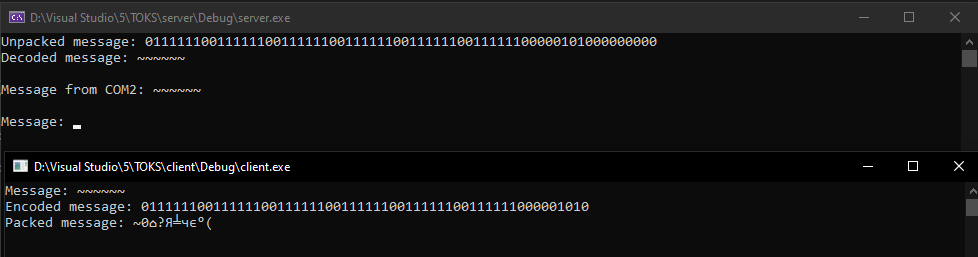
Примеры работы программы:



Бит-стаффинг не требуется.



Так как тильда в двоичном представлении “01111110”, то производится битстаффинг.



string encode(string buffer) {

string encodedData;

int i = 0;

while (i < buffer.size())

encodedData += bitset<8>(buffer[i++]).to\_string();

return encodedData;

}

Метод, предназначенный для кодирования информации в битовую последовательность для последующих операций над пакетом.

string decode(string buffer) {

string decodedData;

for (int i = 0; i < buffer.size() / 8; i++)

{

decodedData += strtol(buffer.substr(i \* 8, 8).c\_str(), NULL, 2);

}

return decodedData;

}

Метод, обратный предыдущему методу. Переводит битовую последовательность в строку.

vector<string> packData(string buffer) {

vector<string> packs;

for (int i = 0; i <= buffer.size() / PACKET\_SIZE; i++)

{

packs.push\_back("");

string packedMsg(buffer.substr(i \* PACKET\_SIZE, PACKET\_SIZE));

int poz = 0;

while (1)

{

poz = packedMsg.find("111111", poz);

if (poz != -1)

{

packedMsg.insert(poz, "1");

poz += 6;

}

else break;

}

while (packedMsg.size() != PACKET\_SIZE + 8) packedMsg.push\_back('0');

int ones = 0;

for (auto digit : packedMsg) if (digit == '1') ones++;

packedMsg.insert(0, flag);

packs[i] = decode(packedMsg);

packs[i].insert(1, to\_string(ones % 2));

}

return packs;

}

Упаковывает данные в пакеты. На этом этапе происходит битстаффинг.

string unpackData(string packedData) {

int parityBit = (int)decode(packedData.substr(8, 8))[0] - 48;

packedData.erase(8, 8);

int poz = packedData.find(flag);

if (poz == -1) return "";

else packedData.erase(0, 8);

int ones = 0;

for (auto digit : packedData) if (digit == '1') ones++;

if (ones % 2 != parityBit) return "";

poz = 0;

while (1)

{

poz = packedData.find("1111111", poz);

if (poz != -1)

{

packedData.erase(poz, 1);

packedData.push\_back('0');

poz += 6;

}

else break;

}

return packedData;

}

Данный метод, распаковывает пакеты, производит дебитстаффинг если он был произведен на передатчике.